A new simulation approach for HPC interconnects

Ronald Luijten (lui@zurich.ibm.com)
IBM Rueschlikon
11 March 2009
Alternative title:

A Journey from Telco Switching to Computer Interconnects…

PRIZMA: Packetized Routing In Zurich's Modular Architecture

Ended 2003

2009
PRIZMA’s Robust Performance

Virtual Output Queuing

Shared output Queueing

Virtual Output Queuing with Output Buffering

- Robust switch performance for extreme traffic conditions
- Memory used for contention resolution only - not for performance
- More scalable solution compared to VOQ - crossbar approach
Measurement on ZRL Regatta w/ Federation

- Hardware Management Console
- One 32-processor SMP
- Gigabit Ethernet Switch (Cisco Catalyst 6513)
- Intranet
- Measurement commands and data
- Synchronization Server
- Interconnection Network (HPS/Federation)
- Switch chip
Regatta CPMD Communication stack

CPMD
Message Passing all-to-all Implementation
Message Passing Interface (MPI) library
Low-level application programming interface (LAPI)
Operating System (AIX 5.3)

Communications Adapter
Interconnection Network

Software
Hardware
Measurement Results (Parallel Efficiency)

- **Measurement config:**
  - IBM MPI
  - LAPI
  - IBM SMA3 adapter
  - user-space mode
  - HPS

- Ideally: parallel efficiency is constant (1.00)

- Strong evidence that **contention in the interconnect causes decrease in parallel efficiency**
- Interconnect only ~20% used

- Suspect effect of interconnect contention on inefficiency in the software layers
OSMOSIS Demonstrator System - Overview

(Optical Shared MemOry Supercomputer Interconnect System)

- 64-ports at 40 Gb/s port speed
- Broadcast-and-select architecture
- Combination of wavelength and space division multiplexing
- Fast switching based on SOAs
Insights gained 3 years after our transition
PERCS phase-2 full-system model

Fat-Tree interconnection network

configureable task-to-node placement via OMNEST configuration file

MPI application trace files
MARS trace replay

Fat-Tree interconnection network

MPI task trace file (recorded or synthetically generated)

configuration task-to-node placement via OMNET configuration

MPI application

trace files

compute node

proc proc ... proc

... proc

system kernel

task task ... task

... task

task0 task1 task2 task3

... taskM

switch module ... switch module ... switch module

Recv Send

MPI semantic action scheduler

MPI task replay module

reduce Allreduce Barrier ... Broadcast

Irecv Wait Isend

computation

trace reader

MPI task trace file (recorded or synthetically generated)

MP1 semantic action scheduler

MPI task replay module

compute node

switch module ... switch module ... switch module

task0 task1 task2 task3

... taskM

IBM Confidential / W. Denzel - PERCS © 2008 IBM Corporation

12
MARS task placement

Fat-Tree interconnection network

LAMMPS sequential task placement
\[ \text{task}[i] \rightarrow \text{node}[i] \]

Traffic [KB]

Source

Destination

0
64
96
128
0
64
96
128

LAMMPS random task placement
\[ \text{task}[i] \rightarrow \text{node}[j \neq i] \]

Traffic [KB]

Source

Destination

0
64
96
128
0
64
96
128

Configurable task placement: \[ \text{task}[i] \rightarrow \text{node}[j] \]

MPI application trace files

configuration file

compute nodes

switch

switch

switch

switch

switch

switch

switch

switch

switch

switch

switch
Good new direction… However

Where do I get good traces?
Tool Enhancement & Integration

MPI application

MPI trace

Dimemas (client)

co-simulation (socket)

Paraver node trace

Venus (server)

Paraver network trace

Paraver

config file

routes

mapping

topology

= new

Venus (server)
WRF application example

- WRF: 256 threads, sim time ~ 2hours
  - Routes with no conflicts
  - Simulation by Venus
  - Load Balanced

- Some threads take longer:
  - Dependency chains
Communication Dependency chains

Communication time: i.e. link is busy

this loads link to 100%
Observation:
Every thread sends down first, except 256, who has nothing to send down, therefore send up first...
Result: Thread 240 gets TWO packets at the same time, attempting to load the link at > 100%...! (destination contention)
Communication Dependency chains

Result:
As 256 starts first, the 1st transmission of 224 has to wait Until 1 transm. Of 256 completes, then second transm. starts
We also study file I/O performance…

Checkpointing on a BG/P machine, using GPFS
Simulation parameters

- **Network**
  - 128 nodes total
  - Three-level fat tree
  - 8-port switches: $32 + 32 + 16 = 80$ switches

- **Nodes**
  - One compute node per IO node (ION)
  - 16 blocks of 4 MB each per node (64 MB per node)
  - ACK size = 512 B

- **Myrinet**
  - Flit size = 256 B
  - Flit duration = 204.8 ns (10 Gb/s)
  - Switch buffer size = 4 KB/port
  - No overhead (raw BW = user BW)
  - 4 KB segments; interleaved in groups of 4 segments by adapter
  - All latencies zero (link, switch, adapter)

- **Three configurations**
  - 92 ION + 36 FS
  - 96 ION + 32 FS
  - 100 ION + 28 FS
92 ION + 36 FS

~ 70 ms

181 ms
96 ION + 32 FS

Block number @ venus.prv_#2

Send bandwidth @ venus.prv_#2

161 ms
Insights gained 5 years after our transition

- Compute vs communicate
- What am I measuring, really?
- If a porsche were your hpc
Literature
